International Journal of Engineering Research
Volume No.5, Issue No.8, pp : 721-723

ISSN:2319-6890)(online),2347-5013(print)
1 August 2016

Logarithmic Multiplier: An Analytical Review

Parvin Akhter[1], Sachin Bandewar[2], Durgesh Nandan[3]
Department of Electronics and Communication Engineering,Sri Satya Sai College of Engineering
(RKDF university) Bhopal, Madhya Pradesh, India [1, 2]
parvin.akhter90@gmail.com[1], sachin.bandewar9@gmail.com[2], prof.durgeshnandan@gmail.com[3]

Abstract: In last four decades, the Logarithmic Number System (LNS) is the most vocative words in the field of arithmetic operations (like addition, subtraction, multiplication, and division). In all arithmetic operations multiplication is most area consuming component, but researchers have analyzed that, LNS has potential to solve this problem. Hence, this paper gives a detailed and meaningful discussion of the evolution of LNS, systematic developments of the LNS multiplier architecture design, highlights the research areas, a further possibility of improvements, their limitations and finally application in the various fields.

Keywords- Arithmetic circuits, antilogarithmic conversion, Operand decomposition, logarithmic conversion, logarithmic multiplication, logarithmic number system, Mitchell method.

1. Introduction

In last four decades, the Logarithmic Number System (LNS) is the most vocative words in the field of arithmetic operations like (addition, subtraction, multiplication, and division) in field of Digital Signal Processing (DSP) applications [1]. Arithmetic operations are the broadly used in the field of image processing (smoothing), DSP (for filtering applications), and speech processing [2]. As, we know that Multiplication is most an area consuming component of arithmetic operations. Researchers have analyzed that, LNS has potential to solve this problem [2]. Ability of Logarithmic multipliers is that it converts multiplication problems into addition [3]. The logarithmic multiplication has mainly distributed in three steps: (1) conversion of binary numbers into the logarithmic numbers, (2) arithmetic operations are performed into the logarithmic domain, and (3) the antilogarithmic conversion of logarithmic numbers [4]. Block diagram of logarithm-based multiplication is shown below in Fig.1. In this paper, various logarithmic multiplication methods and the algorithm have been investigated. Now, the challenge is to make the multiplication efficient regarding hardware architecture as well as accuracy. Rest of paper has been organized as the brief overview of LNS is described in Section 2. Various implementation techniques of logarithmic multiplication have been explored further in Section 3. Section 4 explores the applications of LNS. Finally, the conclusion is concluded in Section 5.

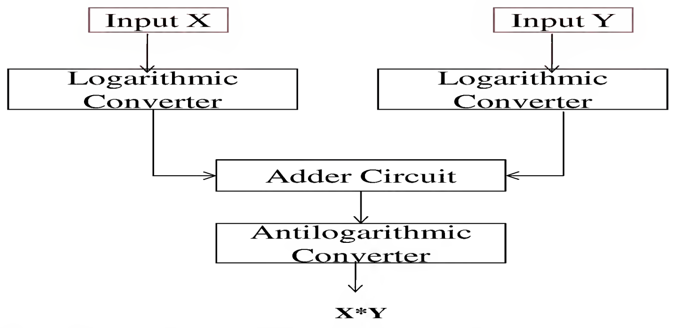

Fig.1. Block diagram of logarithm-based multiplication

2. Overview of The Logarithmic Number System

LNS provide an option in place of binary arithmetic operation for fast computing. The signed logarithmic numbers has been kept in the defined format [5] [6].The IEEE 754 standard sets formats for LNS: 1) Single precision: it has total bit size is 32 bits, and has range of $\approx \pm 1.5\times10^{-39}$ to 3.4×10^{38} 2) Double precision: it has total bit size is 64 bits, and has a range of $\approx \pm 2.8 \times10^{-309}$ to 1.8×10^{-308} [7] .The basic LNS arithmetic operations shown in Table 1.

Table1: Logarithmic Arithmetic Operations

FXP Operation	Logarithmic operation
Z=A×B	Log Z= Log A + Log B
Z=A÷B	Log Z= Log A - Log B
Z=A+B	Log Z= Log A + $\text{Log}_2(1+2^{(\text{Log B- Log A})})$
Z=A-B	Log Z= Log A +$\text{Log}_2(1-2^{(\text{ Log B- Log A})})$

A simple example of multiplication of two logarithmic numbers 8 and 60 is given below [8].
Let A=b' (00001000) =d'16; and B=b' (00111100) =d'60;
Log A =0110.0000; and Log B=0101.11100;
Sum = Log A + Log B
Sum = (0110.0000) + (0101.11100) =1000.11100;
Antilog (Sum) = Antilog (Log A + Log B);
Antilog (Sum) =0000000111100000;
X*Y= Antilog (Sum) =480;
A Logarithm Multiplication Example

3. Various techniques of LNS Multiplication

Conventionally, various methods of LNS multiplication are divided into two broad categories: (1) lookup tables (LUT) based and (2) Mitchell's algorithm based [8].further Mitchell's algorithm can be subdivided into four sub-categories namely as

a) divided approximation b) correction based terms c) operand decomposition and d) MA-based iterative algorithm. Detail descriptions of each method are given in next subsections.

3.1. Logarithmic Multiplication Based on LUT

.In direct lookup table method complete possible values of logarithm are stored in ROM [9], it is also called ROM based method. It is only suitable up to 12 bits more than 20 bits it required long memory space. In some research works, bipartite tables are used for table reduction [10]-[17]. This approach was found faster in speed and yet efficient in resource storage. This method is better to deal with the intermediate length of bits, in the range of 10 to 12 bits. The objective was to identify LUT sizes corresponding to varying logarithmic accuracy levels, trading off accuracy versus memory overhead.

3.2 Mitchell's Multiplication Algorithm

In 1962, Mitchell proposed an algorithm which is based on simple add and shift operation and applicable for multiplication and division [18]. At Mitchell's method, the error lies between zero to 11.1%, the average error is 3.85% [8], and maximum possible error (MPE) is around 11.1% [8].Consider a binary number N for certain define interval range$2^{k+1} < N \geq 2^j$ $and\ k \geq j$. The number N can be express as:

$$N = \sum_{i=j}^{k} 2^i Z_i\ .$$

The final product P is.

$$P_{M.A} = 2^{k_1+k_2}(1 + x_1 + x_2).\ \text{where}\ x_1 + x_2 < 1$$

$$P_{M.A} = 2^{1+k_1+k_2}(x_1 + x_2).\ \text{where}\ x_1 + x_2 \geq 1$$

The error $E_{M.A}$ in Mitchell's algorithm for multiplication is define as

$$E_{M.A} = \frac{(P_{true}) - (P_{approx})}{(P_{true})}$$

In Mitchell's logarithm that uses a piecewise linear curve and producing larger errors was later improved [19]-[26]. In these methods, a conversion method to achieve high accuracy with lower delay and area costs. Hall's algorithm [38] is using all bits in the mantissa for adjustment. Mantissa was divided into four sub-parts and applying linear piecewise approximation. Any part of the mantissa the best linear mean-square coefficients can be determined and evaluated after that developed correction equations for the four intervals with the objective of minimizing the error, which reduced the percentage of maximum error in logarithmic multiplication from 11.1 % in Mitchell's method to 1.3%. Hall's algorithm found high accuracy at the cost of speed, high power consumption, and more hardware complexity [27]. SanGregory's correcting algorithm is small and fast because it uses only mantissa's four MSB for adjustment of concatenated result [19]. SanGregory's proposed two region conversion methods are performed using only combinational logic and requires no multiplications. Abed and Siferd developed correction algorithm that required a trade-off between the accuracy, speed and complexity. It has three different correction strategies based on equations for two, three, and six regions with varying hardware complexity and accuracy. That reduces the maximum percent errors that result from 0.9299 percent, 0.4314 percent, and 0.1538 percent. Juang *et. al.* [22] uses a two region conversion method to achieve high accuracy with less area and complexity of the circuit. It proposed a two region bit level manipulation schemes to achieve high accuracy with the area, time and efficient hardware implementation. Similar approach has been used to achieve accuracy for an antilogarithmic converter. Table 2 give the overall accuracy comparison of all existing logarithm conversion methods regarding a number of partitioned regions, maximum positive error, maximum negative error, error range, maximum positive percent error, maximum negative percent error and percent error range.

Table 4: The Accuracy Comparison of Various Logarithm Conversion Methods

	Number of regions	Maximum positive error	Maximum negative error	Error range	Maximum positive % error	Maximum negative % error	Percent error range
Mitchell [18]	1	0.086	0	0.086	5.791	0	5.791
SanGregory [19]	2	0.0292	-0.028	0.0572	2.503	-1.704	4.207
Abed and Siferd [20]	2	0.0449	-0.0183	0.0632	3.838	-1.076	4.914
	6	0.0132	-0.0129	0.0261	1.216	-0.716	1.932
Juang [22]	2	0.0369	-0.0097	0.0466	2.963	-0.487	3.45
Juang [23]	2	0.0319	0	0.0319	2.337	0	2.337

The correction term based methods analyzed error generated by the Mitchell algorithms and analysis used for a method which improves the accuracy. It can be achieved by adding a correction term either to the final product or the logarithm summation [18], [28]. For Mitchell's Error Correction (MEC) carryover bit from the mantissa part to the integer part determines which one of the following two equations are to be used for correction:

$$P_{M.E.C} = P_{M.A} + 2^{k_1+k_2}x_1x_2.\ \text{where}\ x_1 + x_2 < 1$$

$$P_{M.E.C} = P_{M.A} + 2^{k_1+k_2}y_1y_2.\ \text{where}\ x_1 + x_2 \geq 1$$

Duncan calculated a correction value and added to the logarithm summation. In this approach, the mantissa part of the two input logarithms is partitioned into eight subintervals in steps of 0.125, and a fixed correction term was calculated for each range of the input operands. The operand decomposition is the independent approach of minimizing error, and equally applicable to all previous logarithmic multiplication approaches [8]. For multiplying two n bit binary numbers X and Y at first, the operand X and Y are decomposed into the following four operands A, B, C and D. where decomposed operands are calculated using the following equations:

A = X OR Y,
B = X AND Y,
C = (NOT X) AND Y,
D = X AND (NOT Y).

The product is computed from the decomposed operands using the following property.

X*Y = (A*B) +(C*D)

The operand decomposition approach improves the average error percentage and the error range of Mitchell algorithms. It is equally applicable on all other methods like Divided Approximation based correction and correction term based methods. The iterative logarithmic approximation is based on the correction terms, calculated immediately after the calculation of the product which avoids the comparison of the sum of mantissa with '1.' In this way, high-level of parallelism can be achieved by the principle of pipelining, thus the basic block for multiplication requires less logic resources and increasing the speed of the multiplier with error correction circuits.

4. LNS Multiplication Based Applications

In this section, we mainly focus on the real world applications which are benefit from the use of LNS multiplier has been given. LNS is broadly used in the field of digital signal processing (DSP) applications like Finite Impulse Response (FIR) filtering, Fast Fourier Transform (FFT) and Discrete Cosine Transform(DCT)[1]. Especially logarithmic multiplications are broadly used arithmetic operations in the field of scientific computing, DSP, image processing, speech processing, computer graphics, neural networks, and adaptive systems [2-4]. A Logarithmic Arithmetic Unit (LAU) is proposed for the applications to the mobile 3-D graphics system. By using the LAU, the performance is improved by five times compared with the complex radix-4 method. It is also broadly applicable in video compression in motion vector calculations. The multiplications for the convolution operation during smoothing are performed using Operand Decomposition. Neural network processing comprises a huge number of multiplications.

5. Conclusion

In this paper, we discuss about what is the LNS, how LNS represent, the systematic growth of logarithmic multiplier, utility, benefits, demerits and its applications. This comprehensive study includes the techniques and algorithms used by researchers in LNS multiplication. Based on this review conclude that the iterative logarithmic multiplier is the best choice for designers if accuracy is the main concern because it takes large area, power, and delay. But, the operand decomposition Mitchell's algorithm is the most efficient design in terms of area, and speed that represent hardware-compliant LNS. Future scope for work is that there is the possibility to minimize accuracy and hardware requirement by using hardware minimization techniques.

References

i. I. Kouretas, C. Basetas, and V. Paliouras., "Low-power Logarithmic Number System Addition/Subtraction and Their Impact on Digital Filters." IEEE Transactions on Computers, Vol. 62, No. 11, pp. 2196–2209, Nov. 2013.

ii. A. Klinefelter, J. Ryan, J.Tschanz and B.H. Calhoun, "Error-Energy Analysis of Hardware Logarithmic Approximation Methods for Low Power Applications" IEEE International Symposium on Circuits and Systems (ISCAS), pp.2361-2364, 24-27 May 2015.

iii. L. K. Yu, and D. M. Lewis., "A 30-b Integrated Logarithmic Number System Processor." IEEE Journal of Solid State Circuits, Vol. 26, No. 10, pp. 1433–1440,Oct. 1991.

iv. F.J. Taylor, R. Gill and J. Joseph, "A 20 Bit Logarithmic Number System Processor," IEEE Transactions on Computers, Vol. 37, No. 2, pp. 190-200, Feb. 1988.

v. H. Fu, O. Mencer, and W. Luk, "FPGA Designs with Optimized Logarithmic Arithmetic." IEEE Transactions on Computers, Vol. 59, No. 7, pp. 1000–1006, July 2010.

vi. Jérémie Detrey, and Florent de Dinechin, "A VHDL Library of LNS Operators", Signals, Systems & Computers, The Thrity-Seventh Asilomar Conference on , Vol.2, pp. 2227 – 2231, 9-12 Nov. 2003.

vii. K. Johansson, O. Gustafsson and L. Wanhammar, "Implementation of Elementary Functions for Logarithmic Number Systems." IET Computer & Digital Techniques, Vol. 2, No. 4, pp. 295–304, April 2008.

viii. V. Mahalingam, and N. Rangantathan, Improving Accuracy in Mitchell's Logarithmic Multiplication Using Operand Decomposition, IEEE Transactions on Computers, Vol. 55, No. 2, pp. 1523-1535, December 2006.

ix. E. Swartzlander and A. Alexopoulos, "The sign/logarithm number system," IEEE Transactions on Computers, vol. C, no 12, pp. 1238–1242, December 1975.

x. P. T. P. Tang, "Table-Lookup Algorithms for Elementary Functions and Their Error Analysis." IEEE 10th Symposium on Computer Arithmetic, Grenoble, New York, pp. 232–236, 26–28 June 1991..

xi. M. J. Schulte, and E. E. Swartzlander, "Hardware Designs for Exactly Rounded Elementary Functions." IEEE Transactions on Computers, Vol. 43, No. 8, pp. 964–973, Aug. 1994.

xii. J. E. Stine, and M. J. Schulte, "The Symmetric Table Addition Method for Accurate Function Approximation."Journal of VLSI Signal Processing Systems, Vol. 21, No. 2, pp. 167–177,Feb.1999.

xiii. D. Das Sarma and D. Matula, "Faithful bipartite ROM reciprocal tables," Computers Arithmetic. pp. 17–28, 1995.

xiv. H. Hassler and N. Takagi, "Function evaluation by table look-up and addition," Computers Arithmetic, IEEE Symp., pp. 10–16, 1995.

xv. M. Schulte and J. Stine, "Symmetric bipartite tables for accurate function approximation," Computers Arithmetic Proceedings, pp.175–183, 1997.

xvi. M. J. Schulte, and J. E. Stine, "Approximating Elementary Functions with Symmetric Bipartite Tables," IEEE Transactions on Computers, Vol. 48, No. 8, pp. 842–847, Aug. 1991.

xvii. R. Muscedere, V. Dimitrov, G.A. Jullien, and C.W. Miller, "Efficient Techniques for Binary-to-Multi digit Multi dimensional Logarithmic Number System Conversion Using Range-Addressable Look-Up Tables," IEEE Transactions on Computers, vol. 54, no. 3, pp. 257-272, Mar. 2005.

xviii. J.N. Mitchell, "Computer Multiplication and Division using Binary Logarithms," IRE Transactions on Electronic Computers, Vol. 11, No. 6, pp. 512-517, Aug. 1962.

xix. S.L. San Gregory, R.E. Siferd, C. Brother and D. Gallagher, "Low-Power Logarithm Approximation with CMOS VLSI Implementation," Proc. IEEE Midwest Symp. Circuits and Systems, Aug. 1999.

xx. K.H. Abed, R.E. Sifred, "CMOS VLSI Implementation of a Low-Power Logarithmic Converter," IEEE Transactions on Computers, Vol. 52, No. 11, pp. 1421-1433, November 2003.

xxi. B.G. Nam, H.J. Kim and H.J. Yoo, "Power and Area-Efficient Unified Computation of Vector and Elementary Functions for Handheld 3D Graphics Systems." IEEE Transactions on Computers, Vol. 57, No. 4, pp. 490–504, April 2008.

xxii. T.-B. Juang, S.H. Chen and H.-J. Cheng, "A Lower error and ROM-free Logarithmic Converter for Digital Signal Processing Applications." IEEE Transactions on Circuits and Systems II Vol. 56 (12):pp. 931–935, 2009.

xxiii. T.-B. Juang, P. K. Meher and K.S. Jan. 2011. "High-performance Logarithmic Converters Using Novel Two-region Bit-level Manipulation Schemes." IEEE International Symposium on VLSI Design, Automation and Test, pp.1–4, 25–28 April 2011.

xxiv. Joshua Yung Lih Low and Ching Chuen Jong, "Unified Mitchell-Based Approximation for Efficient Logarithmic Conversion Circuit," IEEE Transaction on Computers, Vol.-64, No.-6, pp. 1783-1797, June 2015.

xxv. K.H. Abed, R.E. Sifred, "VLSI Implementation of a Low-Power Antilogarithmic Converter," IEEE Transactions on Computers, Vol. 52, No. 9, pp. 1221-1228, September 2003.

xxvi. Tso-Bing Juang, Han-Lung Kuo and Kai-Shiang Jan, "Lower-error and area-efficient antilogarithmic converters with bit-correction schemes," Journal of the Chinese Institute of Engineers, 2015.

xxvii. E.L. Hall, D.D. Lynch, and S.J. Dwyer III, "Generation of Products and Quotients Using Approximate Binary Logarithms for Digital Filtering Applications," IEEE Trans. Computers, vol. 19, no. 2, pp. 97-105, Feb. 1970.

xxviii. M.J. Duncan, "Improved Mitchell Based Logarithmic Multiplier for Low Power DSP Applications," IEEE Int'l System on a Chip Conf. (SOCC), pp. 17-20, 2003.